PRÁCTICA 1
IMPLEMENTACIÓN DEL ALGORITMO A*

INGENIERÍA DEL CONOCIMIENTO

Jaime Jiménez Nieto

Introducción

El objetivo de la práctica 1 de la asignatura Ingeniería del Conocimiento es implementar una versión simplificada del algoritmo A* visto en las clases teóricas. Dicho algoritmo consiste en encontrar la ruta óptima en un tablero desde un punto origen y un punto destino o meta, aplicando este algoritmo a un vehículo caminando por un terreno o un avión volando de un punto a otro.
A continuación, se explica de manera breve el contenido de los siguientes apartados:

- **Detalles de la implementación:** Se explica el lenguaje de programación utilizado, así como la estructura de carpetas seguida.
- **Manual de usuario:** Se realiza una explicación detallada de cómo replicar la aplicación en el sistema del usuario de manera local.
- **Ejemplos:** Se muestra una serie de ejemplos a modo de prueba de la aplicación.

Detalles de la implementación

El algoritmo A* se ha desarrollado utilizando **Typescript** y para conseguir la visualización se ha utilizado **Angular** como framework. Todo ello implementado en Visual Studio Code como IDE.

Además de las funcionalidades básicas pedidas en el enunciado de la práctica se ha decidido incluir las siguientes ampliaciones:

- Suponer que determinadas celdas son relativamente peligrosas, pero que pueden atravesarse asumiendo un cierto riesgo. En este caso es necesario introducir un factor de corrección en la función de evaluación heurística.

Estructura del proyecto

Lógica de negocio: Se han creado los siguientes ficheros para implementar la lógica de la aplicación:

- **node.ts:** fichero donde se almacena la clase NodeBoard y toda su información relevante. Esta clase tendrá los siguientes atributos:
    - **position:** Indica la posición de ese nodo con respecto al tablero.
    - **father:** Almacena el nodo antecesor, el nodo padre.
    - **g:** función g(n)
    - **h:** función h(n)
    - **f:** función de coste o f(n)
    - **danger:** Indica si se trata de un nodo peligroso. En caso positivo se le añadirá un + 1 a la función coste.
- **astar.ts:** fichero dónde se aplica el algoritmo A* teniendo en cuenta las funcionalidades explicadas anteriormente. El método algorithm almacenado en el fichero ejecutará el algoritmo y generará la ruta óptima.

Visualización: Para la visualización ha sido falta la creación de varios componentes de Angular, cada uno de ellos ejecutando una única función:

- **board**: componente que proporciona el espacio de navegación, así como los otros componentes.
- **input-board:** componente que se encarga de proporcionar un formulario al usuario para que este indique el tamaño que desea del tablero. Los resultados del formulario serán enviados a través de un emisor propio de Angular al componente padre **board**
- **input-position:** componente que se encarga de proporcionar un formulario al usuario para que este indique las distintas posiciones, tanto la inicial como la de destino. . Los resultados del formulario serán enviados a través de un emisor propio de Angular al componente padre **board**
- **input-obstacles:** componente que se encarga de proporcionar un formulario al usuario para que este indique las distintas posiciones donde se encuentran los obstáculos. Los resultados del formulario serán enviados a través de un emisor propio de Angular al componente padre **board**
- **input-dangerous:** componente que se encarga de proporcionar un formulario al usuario para que este indique las distintas posiciones donde se encuentran los obstáculos. Los resultados del formulario serán enviados a través de un emisor propio de Angular al componente padre **board**.
- **node:** componente que se encarga de dibujar los diferentes nodos del tablero. Este componente almacneará los datos de cada nodo:
    - **posición:** localización del nodo en el tablero
    - **isInit:** determina si el nodo es el nodo inicial
    - **isTarget:** determina si el nodo es el nodo destino
    - **isObstacle:** determina si se trata de un obstáculo
    - **isDangerous:** indica si se trata de un nodo peligroso

Manual de usuario

Para poder ejecutar el proyecto es necesario tener instalado Node.js y Angular en el sistema. Node.js se puede instalar a través de su página web: **https://nodejs.org/en.** Una vez instalado Node.js se ejecuta el comando: npm install -g @angular/cli para instalar Angular

1. Importar el proyecto AStar en Visual Studio Code

2. Instalación de los paquetes necesarios

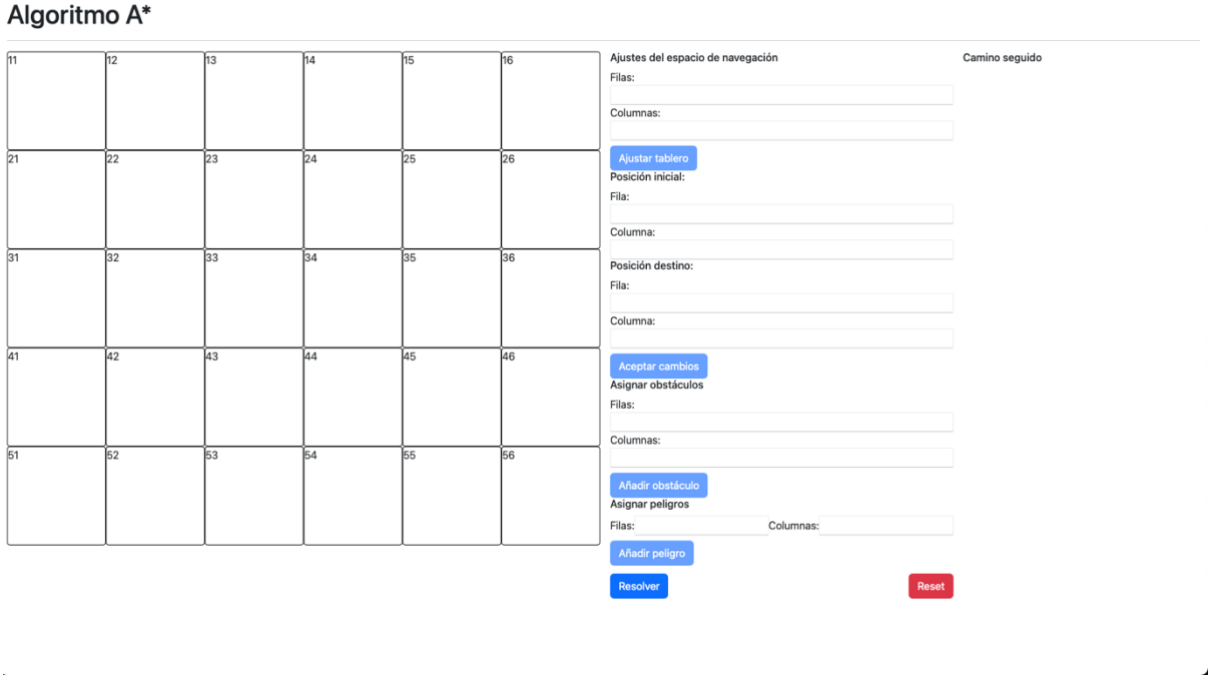
Para poder ejecutar el proyecto es necesario instalar todos los paquetes declarados en el fichero package.json. Para ello se utiliza el comando **npm i.**

3. Arrancar el proyecto

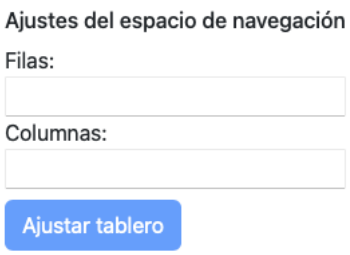
Se ejecuta en la terminal del proyecto el comando **npm start** y se abre en un navegador predeterminado la dirección localhost:4200 para poder acceder a la aplicación.

4. Configuración del espacio de navegación

Una vez arrancado el proyecto aparece la siguiente pantalla:



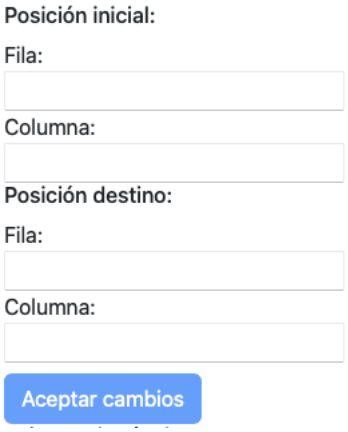
El espacio de navegación que viene por defecto tiene una dimensión de 5 filas y 6 columnas. Esta dimensión puede modificarse a través de uno de los formularios que viene en la parte derecha de la pantalla:



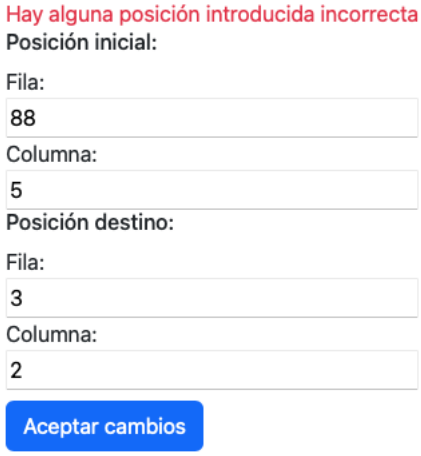
El formulario está diseñado para admitir solo números y si se le introduce un carácter que no pertenezca a esa restricción el botón correspondiente (**Ajustar tablero**) no se verá activo.

5. Asignación de las casillas de inicio y destino

Una vez definido la dimensión del tablero o espacio de navegación se introducen los nodos que se consideran como nodo de inicio o de salida y nodo destino. Para ello se introducen estos campos en el formulario correspondiente.

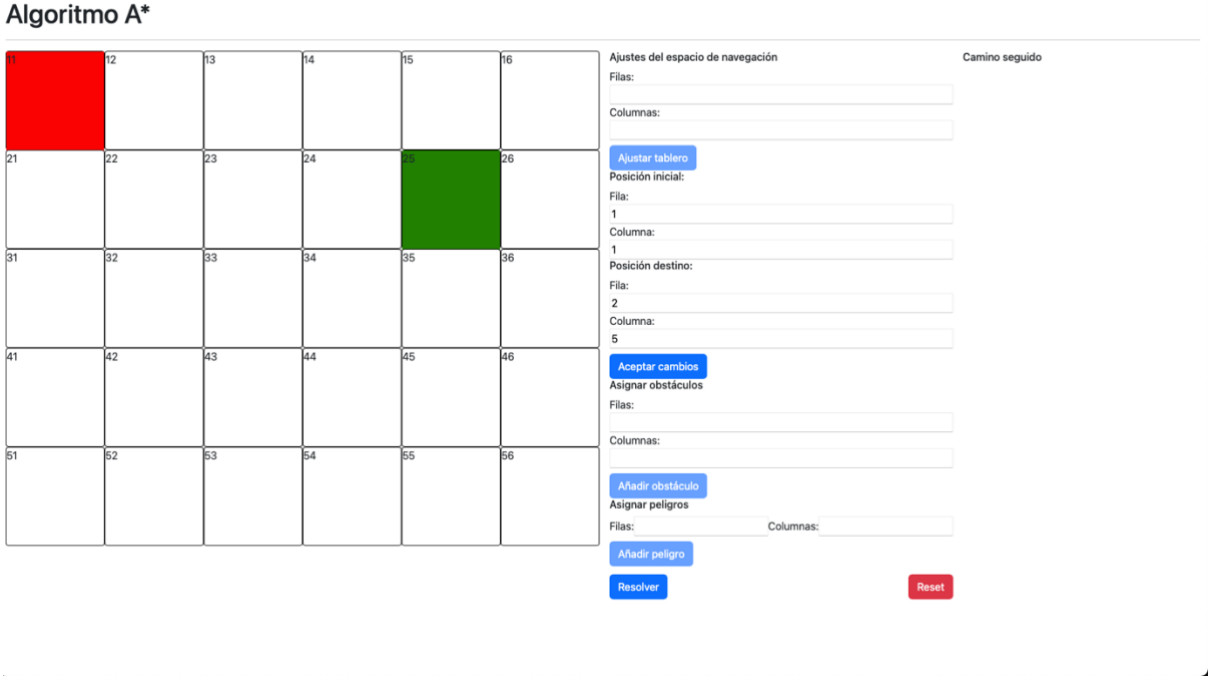


Si la posición de uno de esos nodos no es válida debido a que sobrepasa el tamaño del tablero se le notificará al usuario con un mensaje de error.



El formulario está diseñado para admitir solo números y si se le introduce un carácter que no pertenezca a esa restricción el botón correspondiente (**Aceptar cambios**) no se verá activo.

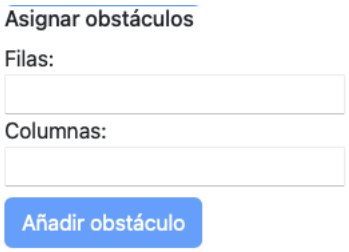
Una vez asignados los nodos inicial y final se tendrá modificado el tablero de la siguiente manera:



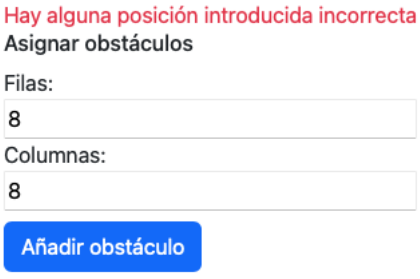
Identificando el nodo inicial como la casilla marcada por el color rojo y el nodo final o destino marcado por el color verde.

6. Asignación de obstáculos

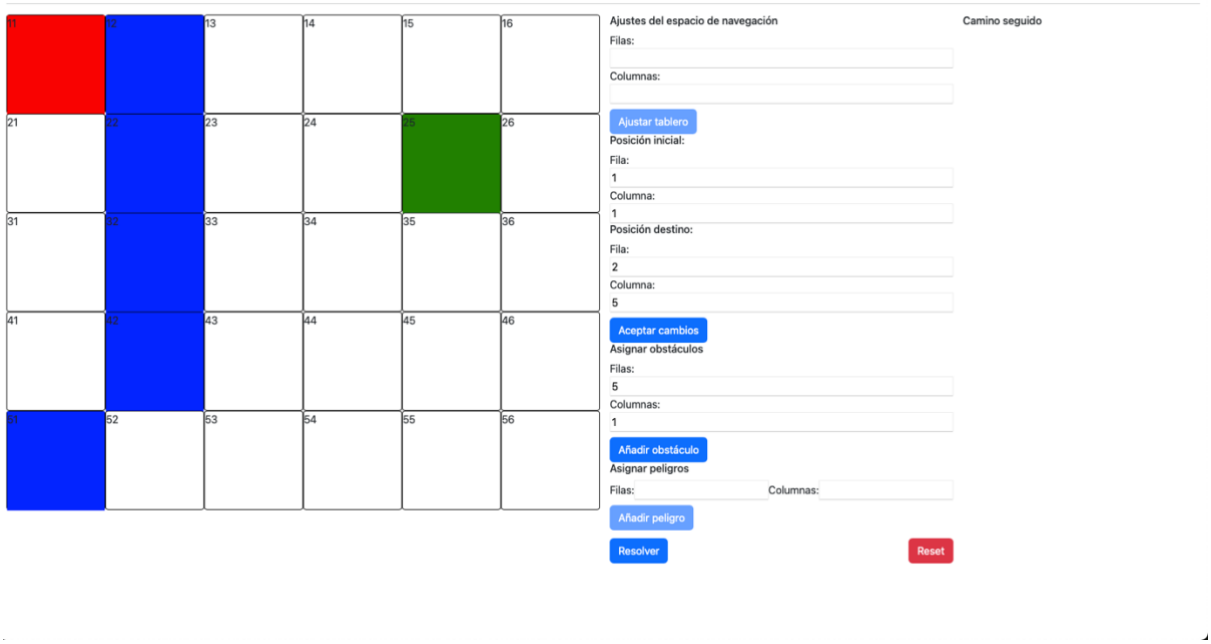
Si se desea se pueden asignar nodos obstáculo al tablero mediante el formulario correspondiente



Si la posición del nodo no es válida debido a que sobrepasa el tamaño del tablero se le notificará al usuario con un mensaje de error.



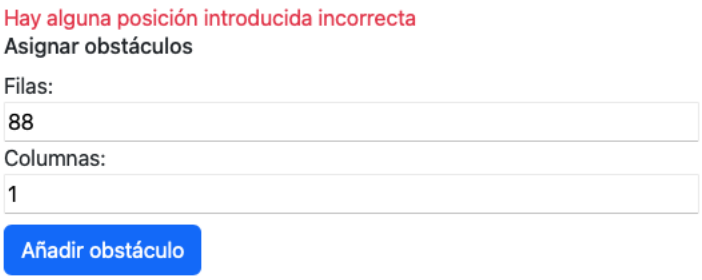
El formulario está diseñado para admitir solo números y si se le introduce un carácter que no pertenezca a esa restricción el botón correspondiente (**Añadir obstáculo**) no se verá activo.
Una vez asignados los nodos obstáculos el espacio de navegación o tablero quedará de la siguiente manera, identificando estos nodos por el color azul



7. Generación de nodos peligrosos

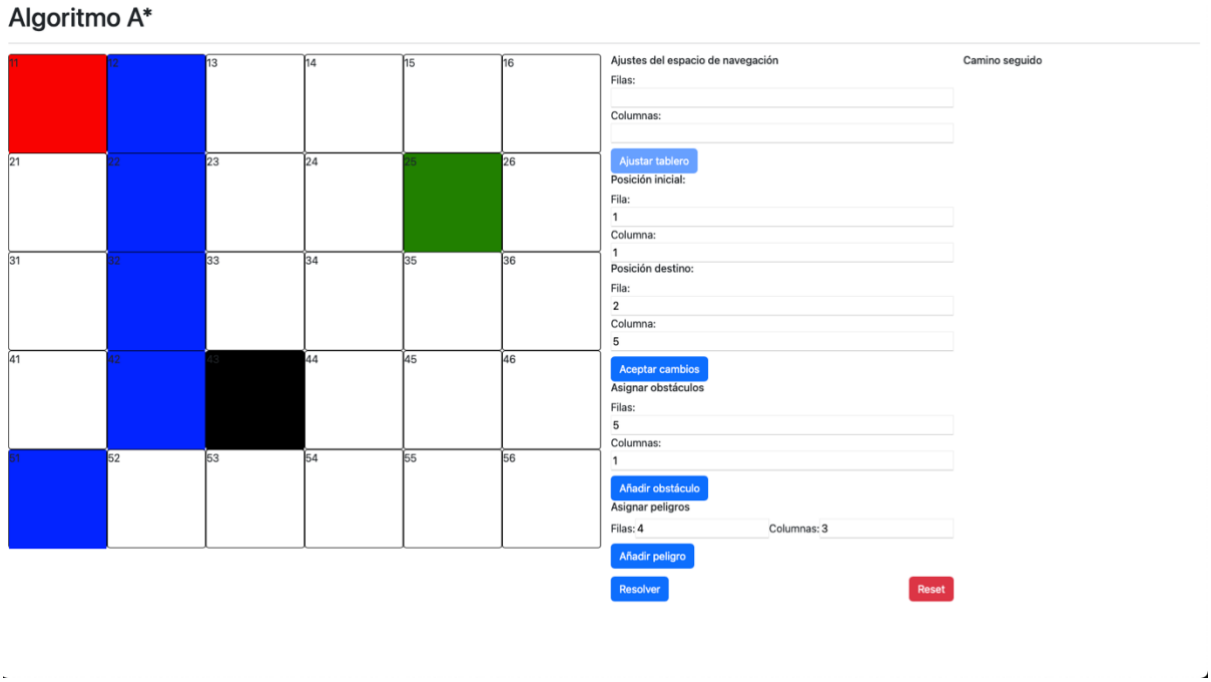
Si se desea se pueden asignar nodos peligrosos al tablero mediante el formulario correspondiente

Si la posición del nodo no es válida debido a que sobrepasa el tamaño del tablero se le notificará al usuario con un mensaje de error.



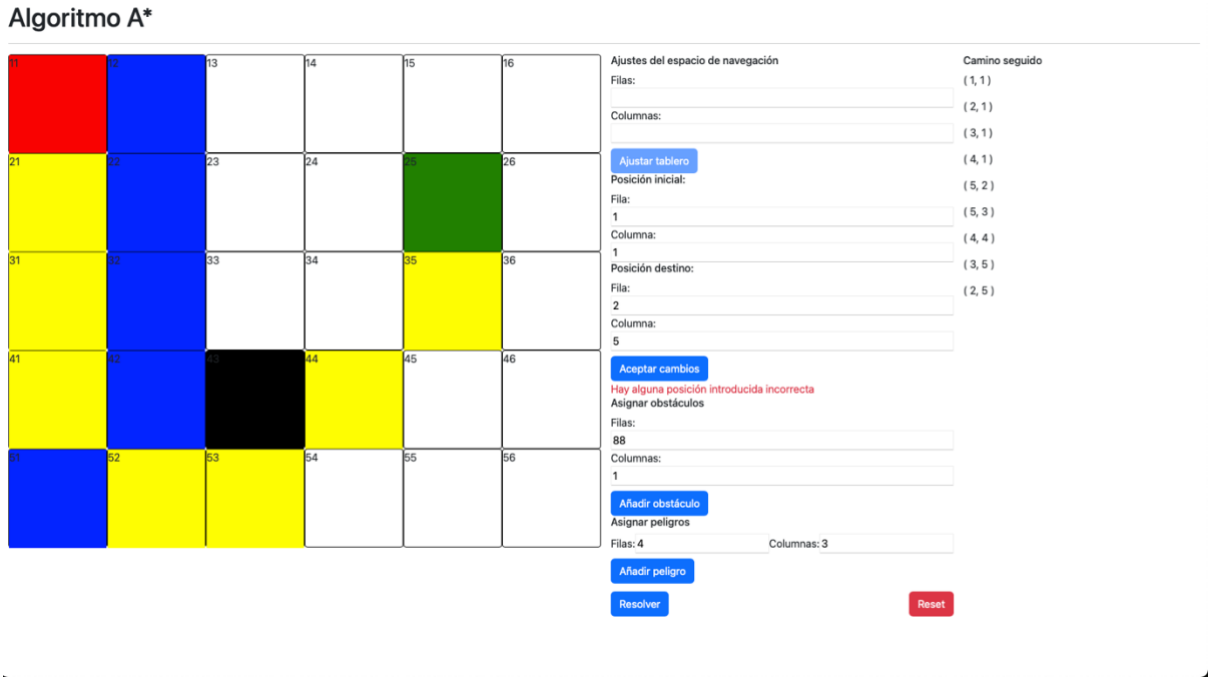
El formulario está diseñado para admitir solo números y si se le introduce un carácter que no pertenezca a esa restricción el botón correspondiente (**Añadir obstáculo**) no se verá activo.

Una vez asignados los nodos peligrosos el espacio de navegación o tablero quedará de la siguiente manera, identificando estos nodos por el color negro.



8. Generación de la ruta óptima

Ya declarados tanto los nodos inicio y destino como los nodos obstáculos y los nodos peligrosos se pulsará en el botón resolver situado debajo de los formularios anteriormente explicados para generar la ruta óptima. Esta ruta marcará los nodos que corresponden a ella con el color amarillo

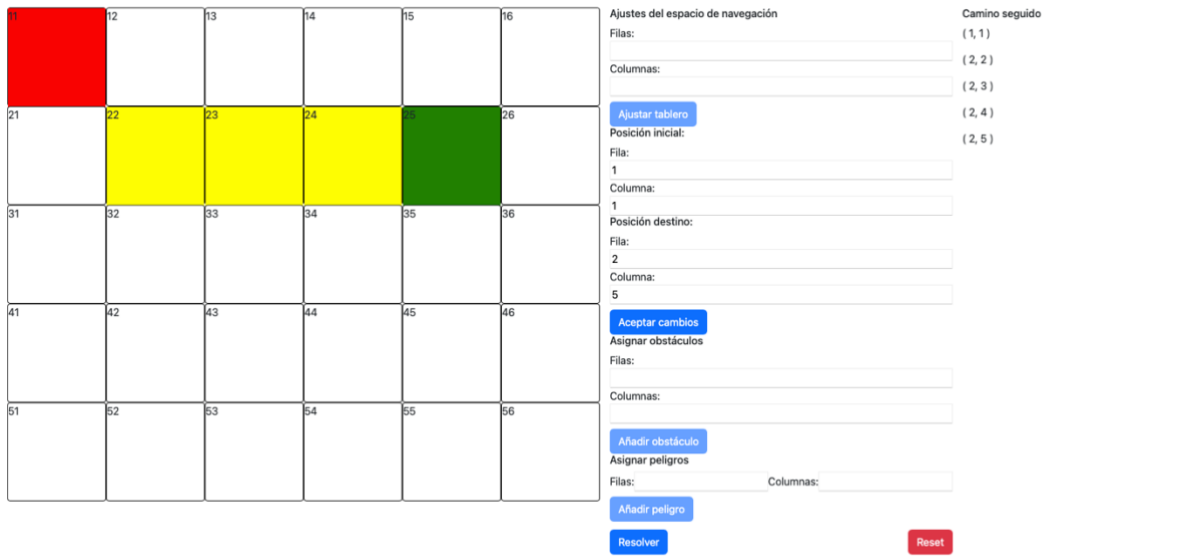


Se ha incluido en la parte derecha de la pantalla el camino que ha seguido el algoritmo para generar la ruta óptima.

Ejemplos

1. Ejemplo 1

Algoritmo A*



2. Ejemplo 2

Algoritmo A*

3. Ejemplo 3

Algoritmo A*